Fall 2019

Modeling Neutron Interaction Inside a 2D Reactor Using Monte Carlo Method

A. S. M. Fakhrul Islam

MODELING NEUTRON INTERACTION INSIDE A 2D REACTOR USING MONTE CARLO METHOD

by

A. S. M. Fakhrul Islam

Bachelor of Science
University of Dhaka, 2009

Master of Science
University of Montana, 2015

Submitted in Partial Fulfillment of the Requirements

For the Degree of Master of Science in

Nuclear Engineering

College of Engineering and Computing

University of South Carolina

2019

Accepted by:

Anthony Scopatz, Director of Thesis

Jamil A. Khan, Reader

Cheryl L. Addy, Vice Provost and Dean of the Graduate School

ACKNOWLEDGEMENTS

My thesis advisor Dr. Anthony Scopatz, my parents and my wife.

ABSTRACT

Scientists and engineers have been working for many years to develop accurate approaches to analyzing nuclear power reactors using computer codes that closely model the behavior of neutrons in a reactor core. The Monte Carlo simulation method is capable of treating complex geometries with a high level of resolution and fidelity to model neutron interactions inside a reactor core. With the requirement of accurate modeling in reactor physics and dynamics and great innovation of computer technology, Monte Carlo method is becoming an ever more powerful tool and receiving rising attention. In this study, Monte Carlo method is used to model nuclear interactions between randomly moving neutrons and the fuel material, cladding material and moderator. The code, QualifyingMC, written using Python language develops the neutron diffusion scenario in a two-dimensional cartesian geometry. To evaluate the performance and accuracy of the simulation, the calculated values of the effective multiplication factor (k_{eff}), a key component in characterizing the breeding property of a fission-reactor system, was compared with reference values calculated with other codes using the same geometry, materials and boundary conditions. A good agreement within a few percent on multiplication factors was obtained. The neutron flux distribution, another important parameter in a fission-reactor system, as a function of neutron energy is also calculated and compared with the Watt distribution function. A reasonable agreement between QualifyingMC and the reference results was obtained.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

a	Collision parameter (unitlesss)
d	Neutron travel distance (cm)
E	Initial neutron kinetic energy (eV)
E'	Kinetic energy of neutron after collision (eV)
I_o	Incident neutron beam intensity (neutron/cm^2/s)
I	Final Neutron beam intensity (neutron/cm^2/s)
k_{eff}	Effective multiplication factor (unitless)
k	Boltzmann Constant (unitless)
l	Neutron mean free path (cm)
p	Pitch (cm)
r_{fuel}	Fuel pin radius/Neutron radial position (cm)
r_{clad_inner}	Inner cladding radius (cm)
r_{clad_outer}	Outer cladding radius (cm)
T	Temperature (K)
v	Velocity (cm/s)
ξ	Random number (unitless)
θ	Neutron angular position (rad)
Σ_f	Fission cross section (cm)
Σ_c	Capture cross section (cm)

Σ_s Scattering cross section (cm)

Σ_t Total cross section (cm)

LIST OF ABBREVIATIONS

A …… Atomic mass number

Al …… Aluminum

ANL ……... Argonne National Laboratory

BNFL …… British Nuclear Fuels Limited

C …… Fuel pin cladding region

CDF ……..... Cumulative Distribution Function

cm ……..…. Centimeter

CPU …… Central Processing Unit

2D …… Two Dimensional

ENDF ……...… Evaluated Nuclear Data File

eV ……...…… Electron Volt

F ……. Fuel region

GHz …….. Gigahertz

gMVP ……..... Global Metabolic Volume Product

H_2O …… Hydrogen Dioxide

HP …….. Hewlett-Packard

IRSN ……... Institut de Radioprotection et de Sûreté Nucléaire

JAERI …… Japan Atomic Energy Research Institute

K …… Kelvin

LANL ……... Los Alamos National Laboratory

LLNL ……. Lawrence Livermore National Laboratory

M …….. Moderator region

MC …… Monte Carlo

MCNP ……... Monte Carlo N-Particle Transport Code

MeV ……... Mega electron volt

MONK …….. Monte Carlo Neutronics Code

MORA …….. Monte Carlo Reactor Analysis

MVP …….. Metabolic Volume Product

N …….. Number of neutrons

NumPy …… Numeric Python

ORNL ……. Oak Ridge National Laboratory

PB ……... Periodic boundary

PDF …….. Probability Distribution Function

PSG ……...…… Probabilistic Scattering Game

QualifyingMC …… Qualifying Monte Carlo code

SCALE …… Standardized Computer Analyses for Licensing Evaluation

U-235 …….. Uranium 235

U-236 …….. Uranium 236

UK ……. United Kingdom

UO_2 ……... Uranium dioxide

US …… United States

USA …….. United States of America

VTT ……Teknologian Tutkimuskeskus

CHAPTER 1

INTRODUCTION

Scientists and engineers have been working for many years to develop accurate approaches to analyzing nuclear power reactors using computer codes that closely model the behavior of neutrons in a reactor core. Two such general approaches and numerical methods employed for the neutron transport simulation [1] are the deterministic and the Monte Carlo method. Deterministic methods [2] involve the solution of differential equations for the transport equation system. It solves the Boltzmann transport equation in a numerically approximated manner everywhere throughout a modeled system and describes the energy dependence of the neutron flux on spatial coordinates of the reactor system. In Monte Carlo simulation methods [3], however, the neutrons are tracked individually from emission to eventual interaction or removal by any nuclear process or leakage. The method is capable of treating complex geometries with a high level of resolution and fidelity. With the requirement of accurate modeling in reactor physics and dynamics and great innovation of computer technology, Monte Carlo method is becoming an ever more powerful tool and receiving rising attention. Some Monte Carlo codes developed by different countries are UK BNFL's MONK[4], France IRSN's MORET[5], Japan JAERI's MVP/gMVP[6] and Finland VTT's SERPENT[7]. Especially in US, at least 5 Monte Carlo codes have been developed, such as LANL's MCNP[8], ANL's VIM[9], ORNL's KENO[10], LLNL's MERCURY[11] and US Navy's MC21[12].

In this study, Monte Carlo method is used to model nuclear interactions between randomly moving neutrons and the fuel material, cladding material and moderator. The code, QualifyingMC, written using Python language develops the neutron diffusion scenario in two-dimensional cartesian geometry. Neutrons are distributed randomly in the fuel region moving through the fissile content uranium dioxide fuel, aluminum cladding material and water moderator. Periodic boundary condition is applied to pincell boundaries. Thus, neutrons are able to leave the reactor without undergoing interaction and appear from the other side of the reactor with the same energy and direction. The initial locations, direction angles and the distance a neutron can travel before undergoing interaction are chosen by random numbers. The diffusion of neutrons inside nuclear reactor resembles the Brownian motion and can be analyzed in the stochastic framework as a random walk. Previous studies show that the use of random walk technique for the analysis of neutron transport process in reactor is quite popular [13]. The assumption is that neutrons will be either absorbed or scattered at which point they will change energy and direction. For the cross-section, energy dependent cross section from ENDF library is used. The cross-section is divided into 13 equal energy groups between $3x10^{-5}$ electron volt and $3x10^{7}$electron volt (eV). This calculational methodology, however, may contain uncertainties caused by several factors such as the multigroup library, multi-dimensionality effects, the "ray-effect", and geometric approximations. Previous studies have shown that the effect of geometric approximations is negligible [14]. Also, a recent study on the effect of quadrature order [15] indicates that the use of a S_8 quadrature set is adequate for these calculations i.e. the "ray-effect" is negligible. Therefore, the uncertainties are caused mainly by the P_3 trun-

cation of the Legendre expansion and/or the energy group structure of the multigroup library. As shown by comparison between multigroup and continuous energy Monte Carlo calculations [14], the multigroup energy Monte Carlo results are within 10% of the continuous energy Monte Carlo results. A better approximation can be made by three-dimensional full-core calculation using continuous energy cross section, but such calculation is beyond the scope of this study.

To evaluate the performance and accuracy of the simulation, the calculated value of effective multiplication factor (k_{eff}), a key component in characterizing the breeding property of a fission-reactor system, was compared with reference values calculated with other codes [16] using the same geometry, materials and boundary conditions [17]. The reference calculation techniques, developed at the VTT Technical Research Centre of Finland, use two codes: the PSG continuous-energy Monte Carlo reactor physics code and MORA full-core Monte Carlo neutron transport code based on homogenisation. The results are then compared to other codes and experimental reference data in the CROCUS reactor kinetics benchmark calculation [17]. The experimental data was then compared to theoretical calculations. Example results provided by four calculation codes; HEXNOD, MCU, HELIOS and BOXER were included in the benchmark specification. Comparing the k_{eff} obtained by these calculations and QualifyingMC, good agreement within a few percent on multiplication factors was obtained. While the reference codes use 20,000 source neutrons in their calculation, QualifyingMC uses up to 2,000 neutrons due to resource limitations. When QualifyingMC is run for higher number of neutrons a better agreement of the

multiplication factor is achieved. The neutron flux distribution, another important parameter in a fission-reactor system, as a function of neutron energy is also calculated and compared with the Watt distribution function [18] and reasonable agreement between QualifyingMC and the reference results was obtained.

CHAPTER 2

BACKGROUND

Increasingly challenging problems in nuclear industry requires higher computational power and more accurate approximations. Monte Carlo has been extensively used in evaluating the performance of new reactor designs [19]. Before the method was developed, simulations tested a previously understood deterministic problem, and statistical sampling was used to estimate uncertainties in the simulations [20]. Monte Carlo simulations invert this approach, solving deterministic problems using a probabilistic analog. An early variant of the Monte Carlo method can be seen in the Buffon's needle experiment [21]. This experiment estimated π by dropping needles on a floor made of parallel and equidistant strips. In the 1930s, Enrico Fermi first experimented with the Monte Carlo method while studying neutron diffusion [22]. The modern version of the Markov Chain Monte Carlo method was invented in the late 1940s by Stanislaw Ulam at the Los Alamos National Laboratory. Since then the method has been widely used in many fields such as nuclear criticality and reactor analysis and radiation shielding calculation, study on probability and statistics of particle. In Monte Carlo methods the computational algorithms rely upon repeated random sampling to obtain numerical results. Their essential idea is using randomness to solve problems that might be deterministic in principle. This is most useful when it is difficult or impossible to use other approaches. Monte Carlo methods are mainly used in three problem classes [23]: optimization, numerical integration, and generating draws from a probability distribution.

They are useful for simulating systems with many coupled degrees of freedom, such as fluids, disordered materials, strongly coupled solids, cellular structures (cellular Potts model, interacting particle systems, McKean-Vlasov processes, kinetic models of gases), calculation of risk in business, evaluation of multidimensional definite integrals with complicated boundary conditions, predictions of failure, cost overruns and schedule overruns, and proved better than human intuition or alternative "soft" methods [24].

In the course of Monte Carlo development, other sophisticated techniques such as “variance reduction” techniques have been introduced to enhance the computational performance of Monte Carlo for higher accuracy and precision [25]. Different variance reduction methods have helped to extend the theory and the sampling technique for improving the efficiency of the simulation runs [26, 27]. The new hybrid Monte Carlo deterministic algorithms significantly improved the capability and performance of the Monte Carlo particle and radiation transport simulations. In the hybrid (Monte Carlo/deterministic) method, the adjoint form of the transport equation is deterministically solved [28, 29]. This solution generates particle weight-window parameters for the optimum sampling application in the Monte Carlo part. For this purpose, deterministic codes and multi group diffusion solver are employed. Some code currently in use includes MCNP in Los Alamos National Laboratory for multiple-purpose particle interaction simulation [30], KENO in National Energy Agency for criticality and flux simulation [31], Serpent in VTT Technical Research Centre of Finland for spatial homogenization and fuel cycle studies [32]. Monte Carlo N-Particle Transport Code (MCNP) had been developed at the Los Alamos National Laboratory since

at least 1957 [33], which later undergone several further major improvements. It is distributed within the United States by the Radiation Safety Information Computational Center in Oak Ridge, Tennessee and internationally by the Nuclear Energy Agency in Paris, France. MCNP is used primarily for the simulation of nuclear processes, such as fission. However, it has the capability to simulate particle interactions involving neutrons, photons, and electrons among other particles. KENO is the primary criticality safety analysis tool in SCALE (Standardized Computer Analyses for Licensing Evaluation). It is a three-dimensional Monte Carlo criticality computer code designed to help a new user understand and use the SCALE/KENO Monte Carlo code for nuclear criticality safety analyses. Serpent is a continuous-energy multi-purpose three-dimensional Monte Carlo particle transport code, currently under development at VTT Technical Research Centre of Finland since 2004 [34]. It was originally known as Probabilistic Scattering Game (PSG) from 2004 to the first pre-release of Serpent 1 in October 2008 [35]. Serpent is used in a wide range of applications from the group constant generation [36] to coupled multi-physics applications, fusion neutronics and radiation shielding [37]. It is a continuous-energy multi-purpose three-dimensional Monte Carlo particle transport code with neutron transport capabilities and able to perform photon transport [38].

CHAPTER 3

THEORY

3.1 RANDOM NUMBER

An important part of any simulation is the ability to generate random numbers. NumPy [39] provides various routines in the submodule random for this purpose. The NumPy package is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation etc. np.random.random is the random function of np.random class which generates random numbers for the given range of 0 to 1. It uses an algorithm, called the Mersenne Twister [40], to generate pseudorandom numbers.

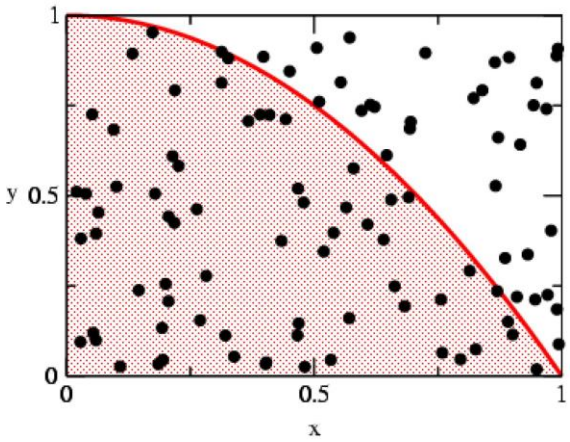


Figure 3.1 Distribution of random numbers to calculate area under a curve function [41]

For the energy distribution of the incoming neutrons and the scattered and fission generated neutron, my model, Qualifying, uses NumPy random numbers for randomly simulating different energy group neutrons. This generates a random level of energy for the scattered neutron which is scattered after the interaction. This ensures the possibility of any possible outcome in the reactor and guarantees the stability of the code and Monte Carlo coded designed reactor in any possible situations. The random function also generates the angular location of initial neutron over the 360 degrees of the nuclear reactor core. It is used to generate random number for radial locations according to the powerlaw distribution [42] of initial neutrons as well. The Monte Carlo methods are used to simulate neutron movement and interactions. The initial locations, direction angles and the distance neutron can go before interaction are chosen by random numbers.

3.2 CROSS SECTION

For the cross-section data, Evaluated Nuclear Data File from the ENDF Database were used. The version of the ENDF is ENDF/B-VIII.0 published in February 2018 in the USA. Cross sectional data for the Fuel (UO_2), cladding material (Al) and moderator was used from this database. Fission cross section data, scattering cross section data and capture cross sectional data were taken. The energy range for these data starts from $3x10^{-5}$ electron volt to $3x10^{7}$electron volt (eV). Cross sectional data for 13 different energy groups were taken by dividing this energy range into 13 equal (linear) groups. Finding the values of these different cross-sectional data, elastic and inelastic scattering cross sectional values were added to determine the total scattering cross sectional data. Again, (n, gamma) cross

section data for these energy groups were used to achieve the cross-sectional data for capture.

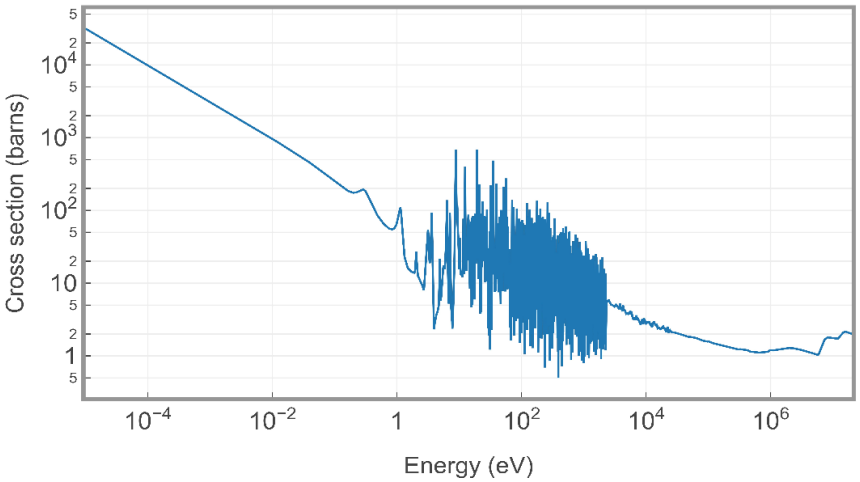


Figure 3.2 Fission cross section for Fuel [43]

As the unit of the values were in barns [10^{-24} cm^2], the units were converted to cm^2 by multiplying each of them by 10^{-24}. Now these microscopic cross section data need to be converted into macroscopic cross section data. In order to that the microscopic cross section data is multiplied by the number density for each element and material. This macroscopic cross section data is used for the coded calculation and simulation. Figures 3.2, 3.3 and 3.4 are original plots for different energy groups of cross-sectional data collected from the ENDF library. In these figures, scattering cross section, fission cross section and capture cross section data are plotted for the fuel material.

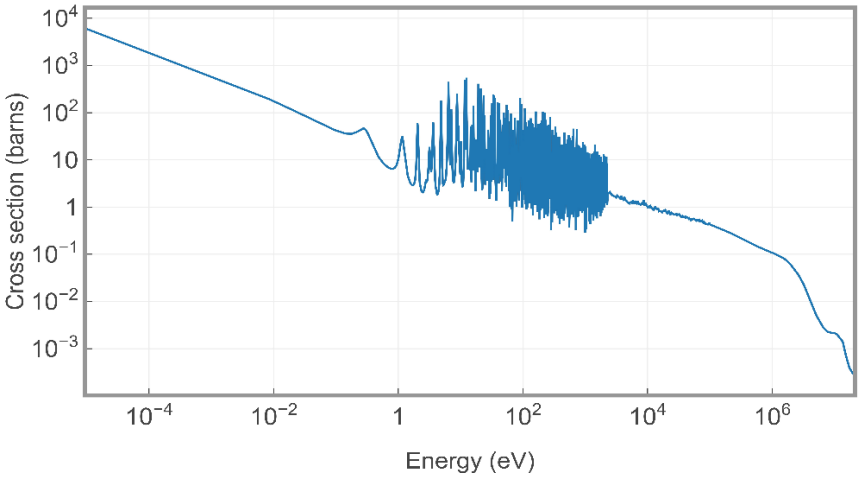


Figure 3.3 Capture cross section for Fuel [43]

The data in these figures are plotted in the units of barns against the scale of eV for the energy. For the benefit of calculation, we these units were converted to 1/cm, the unit for the macroscopic cross section.

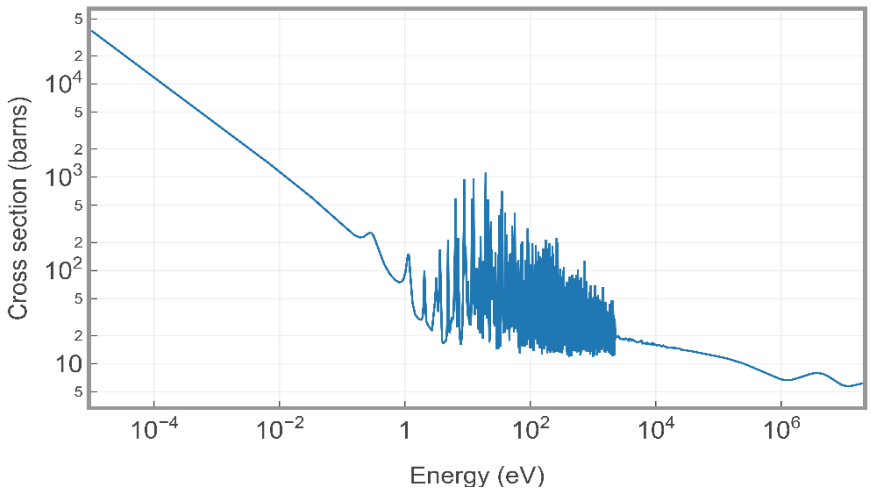


Figure 3.4 Scattering cross section for Fuel [43]

3.3 NEUTRON INTERACTION

Fission reactions are caused by neutrons inside fission reactor. When a neutron hits the fuel nucleus, for example U-235, the neutron is absorbed and an isotope U-236 is formed. The U-236 is invariably in its excited state and must de-excite. We know one possibility of de-excitation is fission, and the entire reactor program depends on this possibility. There are different types of interaction reactions that take place during a reactor operation. Of them, fission, capture, elastic scattering, inelastic scattering are the main ones.

$$ {}^{0}_{1}n + {}^{A}_{Z}N \rightarrow {}^{A+1}_{Z}N^{*} $$

Neutron scattering
- Elastic scattering (n, n)
- Inelastic scattering (n, n')

Neutron absorption
- Radioactive capture (n, γ)
- Fission
- Charge particle emission (n, p) (n, α)
- Neutron emission (n, 2n)

Scattering interactions always produces a single neutron after the interaction. In the interaction, the neutron and the nuclide collide and share a part of their kinetic energies. Then they rebound with speeds different from the original speeds, such that the 'total kinetic energy' before and after the collision remains the same. If the nucleus is stationary before collision, it will gain energy from the neutron and start moving, and the neutron gets slowed down due to loss of kinetic energy. However, the residual nucleus is not excited but is in its ground state. These types of interactions are known as the elastic interactions. Hence, elastic scattering represents scattering where both energy and momentum are conserved.

On the other hand, inelastic scattering takes place when energy and momentum are not conserved. The lost energy and momentum go into exciting the inertial energy state of the nucleus. The neutron and the nuclide collide and rebound with speeds different from the original speeds, but the rebounding nuclide is left in an excited energy state. Hence the 'total kinetic energy' after the collision is less than that before the collision, and this difference accounts for the energy of excitation. If the nucleus is stationary before collision, the neutron must have kinetic energy exceeding the excitation energy, so that such a reaction is possible. Hence inelastic scattering is said to be a threshold reaction, the threshold being the minimum kinetic energy of the neutron required for the reaction to be possible. The excited nucleus subsequently falls down to a lower energy state by emitting gamma radiation. Heavy nuclides have lower thresholds than light nuclides. Though the probability of inelastic scattering is generally lower than elastic, the energy loss to the neutron is higher in an inelastic collision. Inelastic scattering in heavy nuclides degrades fission neutron energies heavily.

In the case of capture, the neutron is absorbed by the target nucleus to form the next higher isotope (of mass A+1), in an excited state of energy. The new isotope de-excites by emitting gamma rays. The neutron is thus lost in this reaction. This is often known as 'radiative capture.'

Fission reactions, on the other hand, are one of the most important reaction upon which the present-day nuclear energy program depends. Nuclear fission is a phenomenon in which a heavy nucleus, splits into two smaller nuclei, called the fission fragments,

mostly of unequal masses: one often with nearly half the mass as the other, and rarely of equal masses. This reaction gives off a large amount of energy and emits two or more neutrons, and gamma rays. When a neutron hits a heavy nuclide like U-235, the neutron gets absorbed in the heavy nuclide that gets energetically agitated (or excited). If the new energy state of the heavy nuclide is sufficient for it to split, then it can split to cause fission. The neutrons produced in fission are fast, with an average energy of 2 MeV. It must be noted that the fission fragments themselves are in excited state, and they de-excite generally by beta, gamma and neutron emissions. The neutron emitted during fission are called prompt neutrons, and those emitted by the fragments after a delay are called delayed neutrons. Similarly, prompt and delayed gammas are also emitted. About 80 % of the energy released in fission is carried away by the fission products (and the rest by the other particles), which in turn transfer the energy to the surroundings, making the energy recoverable. Some energy is carried away by particles known as neutrinos**,** which are charge less and light, do not interact with any material, and hence their energy is not recoverable. As a fission caused by a neutron involves production of further neutrons, a fission chain reaction becomes possible, and such a chain is ensured in the design of a reactor. The nuclear energy released in fission is about a million times the chemical energy released in burning a block of coal of equal mass.

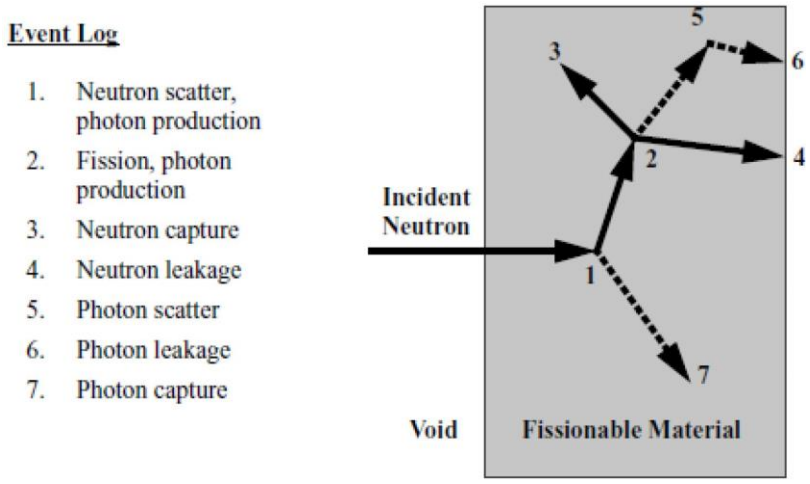


Figure 3.5 Neutron path in Monte Carlo code [41]

There are many other neutron interaction reactions that happen in a nuclear reactor. Here, Fission, Capture and Scattering interactions will be covered to calculate and simulate the neutron interaction within the periodic boundary of a nuclear reactor.

3.4 MEAN FREE PATH

When looking at the transmissions of neutrons through matter within a reactor, the mean free path helps determine the average distance that a neutron travels between each collision. The mean free path is derived from the probability function, $p(x)$ (unitless), shown in Equation 1, in which it determines the probability that the first collision for the neutron will occur in dx (cm) in the neighborhood of x. The mean free path, λ (cm), is then obtained by integrating the probably function with respect to x from 0 to infinity as shown in Equation 2.

$$p(x)dx = e^{-\sum_t x} \times \sum_t dx$$

$$= \sum_t e^{-\sum_t x} dx \quad (1)$$

$$\lambda = \int_0^\infty xp(x)dx = \sum_t \int_0^\infty xe^{-\sum_t x} dx$$

$$= \frac{1}{\sum_t} \quad (2)$$

The mean free path is utilized in many different applications to help determine moderator thickness of a shield. The mean free path relies on both the energy of the neutron and the type of material it passes through. When a neutron collides, it loses energy and its mean free path is also affected.

3.5 NEUTRON COLLISION THEORY

Although understanding the distance between each collision is important, it is also important to understand what occurs during the neutron interaction. All neutrons produced by fission are born as fast neutrons. These fast neutrons slow down by collisions with the nuclei which mostly occur in the moderator of the reactor. Figure 3.6 displays what occurs when a neutron is elastically scattered from a nucleus at rest. From the figure, a probability between the initial kinetic energy, E (MeV), and the kinetic energy after collision, E' (MeV), can be found. Equation 4 is the resulting probability that the kinetic energy will drop from the initial to the final as a result from the collision. Equation 3 is utilized to relate the initial and final energies for a collision, where α (MeV) is the collision parameter, the smallest fractional energy that a neutron can have after collision. The collision parameter

is shown in Equation 5 where A is the atomic mass number. Through this process it is assumed that this is uniform isotropic scattering.

$$E' = \left[\frac{(1+\alpha)+(1-\alpha)cos\theta_c}{2}\right] \times E \tag{3}$$

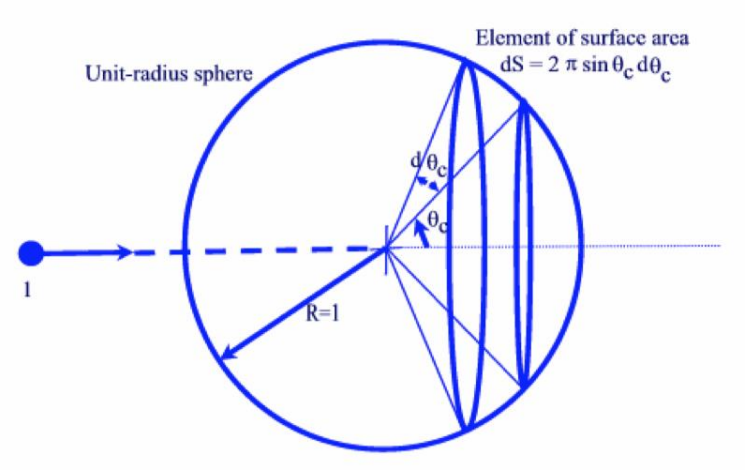


Figure 3.6 Illustration of elastic scattering off a nucleus at rest [42]

$$P(E')dE' = \frac{Surface\ dS}{Total\ unit\ surface}$$

$$= \frac{2\pi sin\theta_c d\theta_c}{4\pi \times 1^2}$$

$$= -\frac{d(cos\theta_c)}{2} \times \frac{dE'}{dE'}$$

$$cos\theta_c = \frac{2}{1-\alpha} \times \frac{E'}{E} - \frac{(1+\alpha)}{(1-\alpha)}$$

$$\frac{d(cos\theta_c)}{dE'} = \frac{2}{(1-\alpha)E}$$

$$P(E')dE' = -\frac{1}{(1-\alpha)E} \times dE'$$

$$P(E') = \frac{1}{(1-\alpha)E} \tag{4}$$

$$\alpha = \left(\frac{A-1}{A+1}\right)^2 \tag{5}$$

Using Equation 5, the average energy of a neutron after one collision can be found as shown in Equation 6. This average energy falls between E and αE since those two, in terms of words, are respectively the maximum energy and the minimum energy resulting from a collision. Figure 3.7 shows this probability in graphical form.

$$\bar{E}' = -\int_{E}^{\alpha E} E' \frac{1}{(1-\alpha)E}\, dE'$$

$$= \frac{E^2 - \alpha^2 E^2}{2} \frac{1}{(1-\alpha)E}$$

$$= (1-\alpha)\frac{E}{2} \tag{6}$$

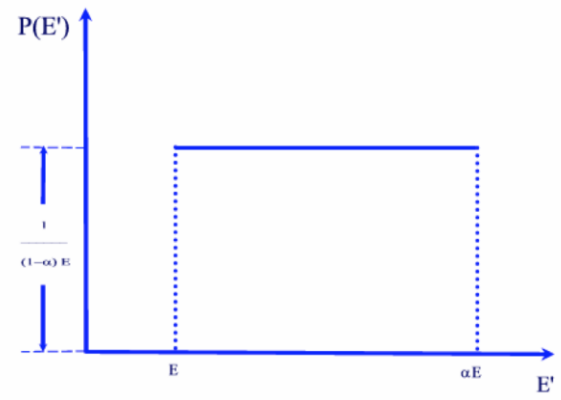


Figure 3.7 Probability of scattering between max. and min. energy after collision [42]

3.6 NEUTRON MODERATOR

Although neutron collision from elastic scattering can slow down the neutrons, a quicker method that is used is the neutron moderator. A moderator is used to slow down the neutrons going from fission to the thermal energies. A moderator is needed because at the thermal energies when the neutrons are slower, the probability of fission U-235 becomes very high. Figure 3.8 shows how the moderator is used, in this scenario it is H_2O, to slow down the neutrons from colliding into each other. To have an effective moderator, it is important to have a high elastic scattering cross-section and a high average logarithmic energy decrement.

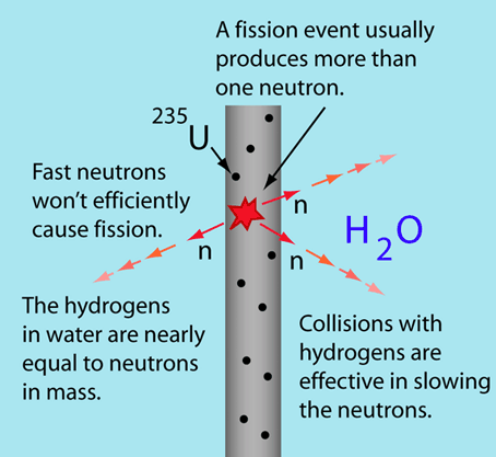


Figure 3.8 Demonstration of how the neutron moderator works [45]

3.7 MAXWELLIAN-BOLTZMANN DISTRIBUTION

Neutron moderators are mainly used in thermal reactors in which the reactor uses thermal neutrons to sustain the chain reaction. Neutrons in the thermal region achieve thermal equilibrium with the atoms of the moderator. These neutron's energies are distributed using Maxwellian-Boltzmann distribution which is shown in Equation 10, where T is the

temperature of the moderator and k is the Boltzmann constant which is 8.52×10^{-5} eV/K. The thermal spectrum is defined in Equation 11 where n_0 is the total thermal neutron density. Figure 3.9 graphically shows the thermal spectrum in relation to the rest of the reactor neutron spectrum.

$$M(E) = \frac{2\pi E^{1/2}}{(\pi kT)^{3/2}} e^{\frac{-E}{kT}} \tag{7}$$

$$\phi(E) = n_0 v M(E) = \sqrt{\frac{2E}{m}} \times n_0 \times \frac{2\pi E^{1/2}}{(\pi kT)^{3/2}} e^{\frac{-E}{kT}} \tag{8}$$

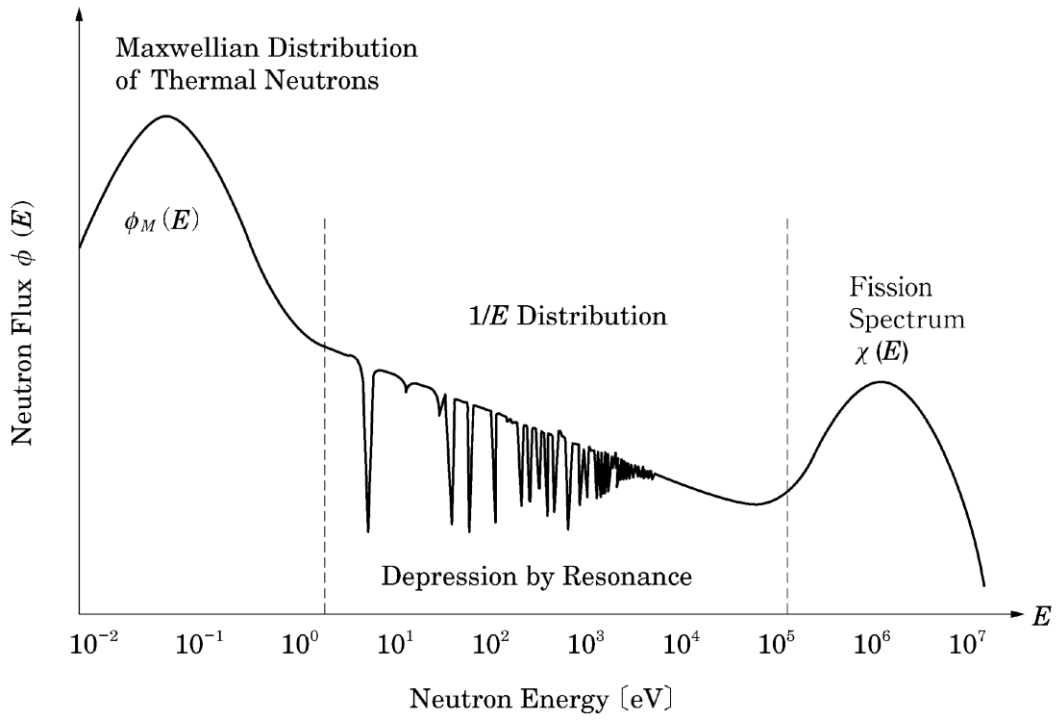


Figure 3.9 Thermal reactor neutron spectrum [45]

Prompt neutrons are born at energies between 0.1 MeV and 10 MeV. The average prompt neutron energy is about 2 MeV. Fast reactors have a neutron energy spectrum that has the same shape as the prompt neutron energy spectrum. Thermal reactors have a neutron energy spectrum that has two pronounced peaks, one in the thermal energy region where the neutrons are in thermal equilibrium with the core materials and another in the fast region at energies where neutrons are produced. The flux in the intermediate region (1 eV to 0.1 MeV) has a roughly 1/E dependence. The neutron flux spectrum for the fast energy region of a thermal reactor has a shape similar to that of the spectrum of neutrons emitted by the fission process. The reason for the 1/E flux dependence at intermediate energy levels in a thermal reactor is due to the neutrons' tendency to lose a constant fraction of energy per collision. Since the neutrons lose a greater amount at the higher energies, the neutrons tend to "pile up" at lower energies where they lose less energy per collision. The neutron flux spectrum for the slow region of a thermal reactor contains a peak at the energy where the neutrons are in thermal equilibrium with the atoms of the surrounding materials.

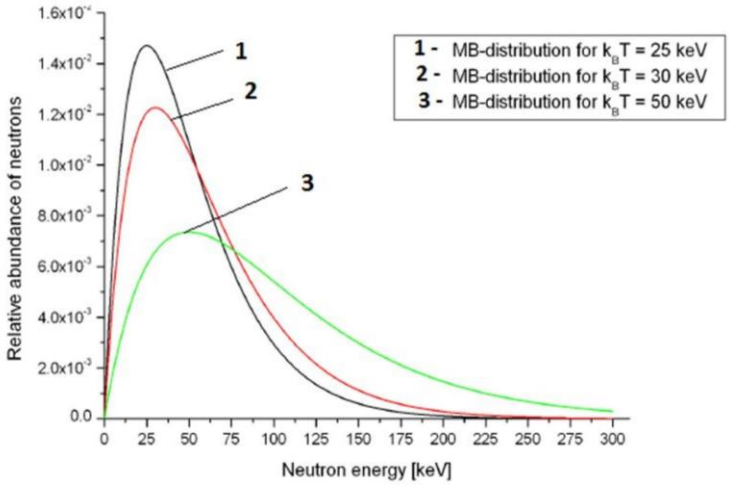


Figure 3.10 Relative abundance of neutrons at different energy and temperatures [46]

3.8 EFFECTIVE MULTIPLICATION FACTOR

Effective multiplication factor, k_{eff} (unitless), is the ratio of the neutrons produced in one generation to neutrons lost in previous generation.

$$k_{eff} = \frac{Neutrons\ Produced}{Neutrons\ Lost} \tag{9}$$

Neutrons are only produced by fission reaction and lost by leakage and absorption. The remaining neutrons either are absorbed in non-fission reactions or leave the system without being absorbed. The value of k_{eff} determines how a nuclear chain reaction proceeds:

k_{eff} < 1 (subcriticality): The system cannot sustain a chain reaction, and any beginning of a chain reaction dies out over time. For every fission that is induced in the system, an average total of 1/(1 − k) fissions occur.

k_{eff} = 1 (criticality): Every fission causes an average of one more fission, leading to a fission (and power) level that is constant. Nuclear power plants operate with k = 1 unless the power level is being increased or decreased.

k_{eff} > 1 (supercriticality): For every fission in the material, it is likely that there will be "k" fissions after the next mean generation time. The result is that the number of fission reactions increases exponentially.

In a nuclear reactor, k_{eff} will actually oscillate from slightly less than 1 to slightly more than 1, due primarily to thermal effects. This leaves the average value of k_{eff} at exactly 1. In my model, I used a simple case of neutrons causing chain reaction without accounting for thermal effect.

CHAPTER 4

METHOD OF ANALYSIS

4.1 GEOMETRY AND BOUNDARY CONDITIONS

The Monte Carlo methods are used to simulate neutron movement and interactions. The initial locations, direction angles and the distance neutron can go before interaction are chosen by random numbers. QualifyingMC is made to demonstrate criticality calculation for simplified 2D pincell geometry by using Monte Carlo methods for 13 group cross sections. Geometry and boundary conditions are shown below in Figure 4.1.

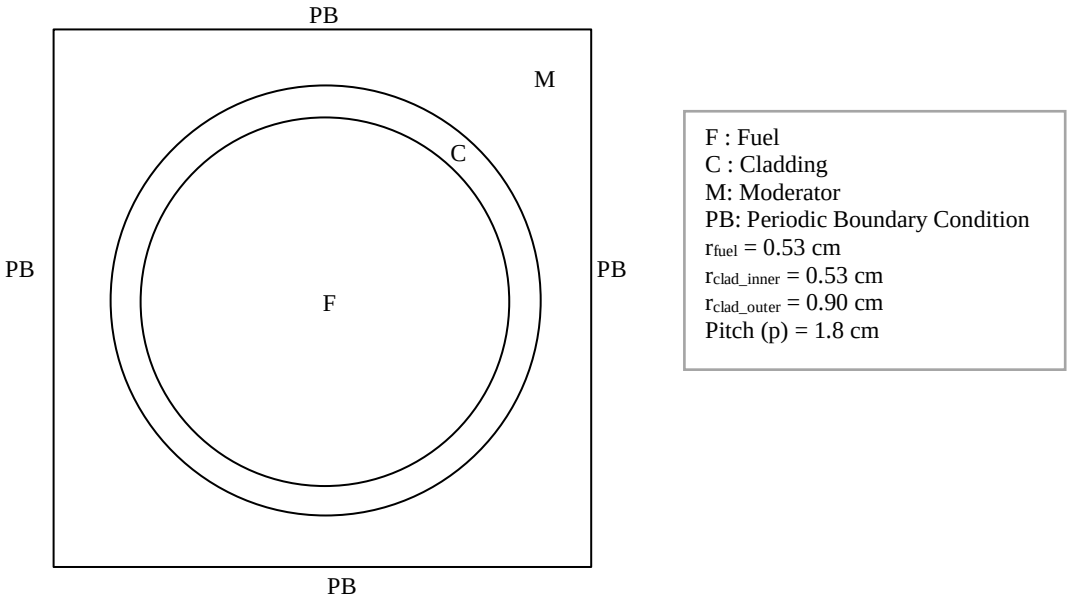


Figure 4.1 The 2D geometry used in QualifyingMC code.

The neutrons are initially distributed uniformly in fuel region. The fissile content Uranium-dioxide (UO_2) is defined as fuel, Aluminum (Al) is defined as cladding material, and Water (H_2O) is defined as moderator. The energy dependent cross sections of these materials are used for calculations. The periodic boundary condition is applied to pincell boundaries, so neutrons are able to leave the system without doing any interaction and appear at the opposite boundary with same direction. The ultimate purpose of calculations done by QualifyingMC is to calculate effective multiplication factor (k_{eff}) for a specific number of neutrons.

4.2 SAMPLING INITIAL NEUTRON LOCATIONS

The initial neutron locations are chosen by using random numbers (ξ) between 0 and 1.

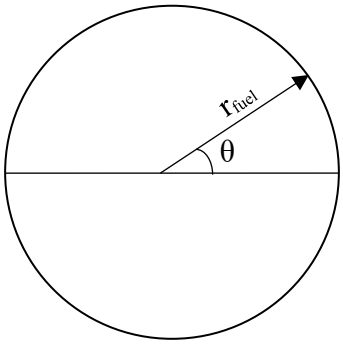


Figure 4.2 Defining Initial location of neutron

To find angular position, theta θ (rad) is sampled uniformly over 2π

$$\theta = 2\pi\xi$$

To find radial position, r is sampled over r_{fuel} by considering power law distribution because of radial effect.

4.3 SAMPLING NEUTRON DIRECTION AND INTERACTION DISTANCE

The direction angle theta (θ) is sampled uniformly over 2π. The distance neutron can go before interaction is related to mean free path of a neutron. To sample it correctly, we need to find cumulative distribution function (CDF) from its probability distribution function (PDF) as below. The probability distribution function of a neutron can be found by using its survival probability.

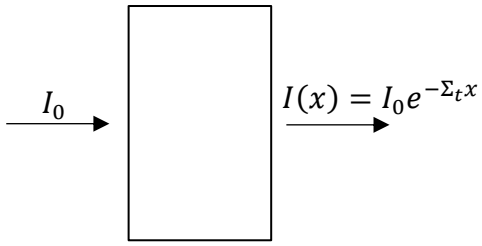


Figure 4.3 Interaction of neutron with a medium

The survival probability, which is the probability for not being interacted in distance x, is $e^{-\Sigma_t x}$.

Then the interaction probability in distance x becomes $\Sigma_t e^{-\Sigma_t x}$

The PDF is $pdf(x) = \Sigma_t e^{-\Sigma_t x}$

Then the CDF is calculated by integrating PDF between 0 and x

$$cdf(x) = \int_0^x \Sigma_t e^{-\Sigma_t x'} dx' = 1 - e^{-\Sigma_t x} \quad (10)$$

CDF is the distribution function to be used for random numbers to sample interaction length d

$$cdf(x) = \xi = 1 - e^{-\Sigma_t x} \quad (11)$$

Then, the interaction length is sampled by following equation

$$d = x = -\frac{1}{\Sigma_t} \ln(\xi) \quad (12)$$

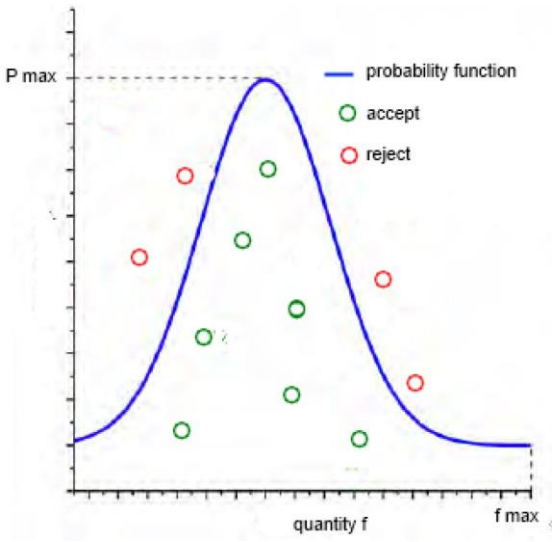


Figure 4.4 Probability distribution function [41]

4.4 CHECKING REGION CHANGE

QualifyingMC stores each neutron position even if it is not interacted, which makes it easy to check whether neutron is leaving a region and entering another one. The easiest way to check region change is calculating all distances in neutron direction to all boundaries and then compare it with the interaction length. If the interaction length is bigger than the closest distance, then the neutron will be moved without being interacted to that surface which has the closest distance.

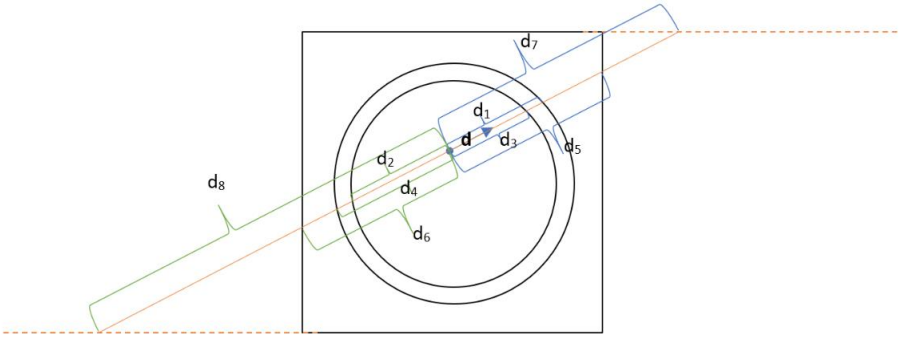


Figure 4.5 Distances to the boundaries

As can be seen in Figure 4.5, there are 8 different distance value for 8 different intersection. These distances are calculated line equations for circles and boundaries. According to neutron direction, d_1, d_3, d_5, d_7 are forward distances and have positive sign. On the other hand, d_2, d_4, d_5, d_6 are backward distances and have negative sign. Among these distance values, the smallest and positive one will be picked and compared to interaction length d to decide whether neutron is leaving the region or not.

It can be assumed that neutrons are travelling in a line with direction angle θ, then the distance values are calculated by using following equations. For circle surface, the intersection points can be calculated geometrically.

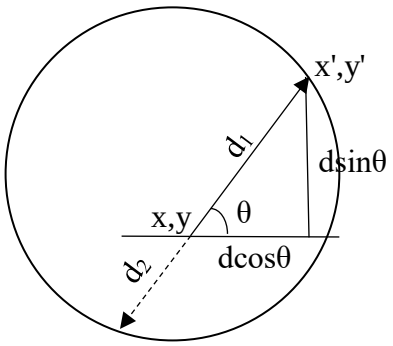


Figure 4.6 Calculating the position of neutron

$$x' = x + d\cos\theta$$

$$y' = y + d\sin\theta$$

$$(x + d\cos\theta)^2 + (y + d\sin\theta)^2 = r^2$$

$$x^2 + 2xd\cos\theta + d^2\cos^2\theta + y^2 + 2yd\sin\theta + d^2\sin^2\theta = r^2$$

$$(\sin^2\theta + \cos^2\theta)d^2 + (2x\cos\theta + 2y\sin\theta)d + (x^2 + y^2 - r^2) = 0$$

$$d^2 + (2x\cos\theta + 2y\sin\theta)d + (x^2 + y^2 - r^2) = 0 \quad (13)$$

This is quadratic equation with coefficients, $a = 1,\ b = 2x\cos\theta + 2y\sin\theta\,,\ c = x^2 + y^2 - r^2$

$$\Delta = b^2 - 4ac$$

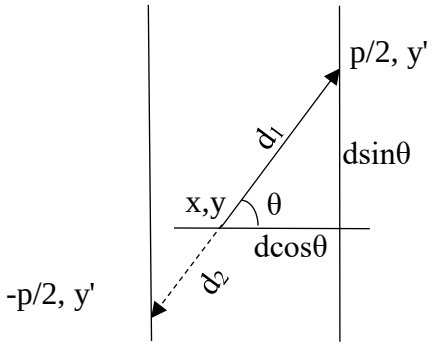
$$\Delta \geq 0 \ \rightarrow \ \text{intersection with the circle surface}$$

$$\Delta < 0 \ \rightarrow \ \text{no intersection with the circle surface}$$

the roots of this equation are d_1 and d_2,

$$d_{1,2} = \frac{-b \mp \sqrt{\Delta}}{2a} = \frac{-b \mp \sqrt{b^2 - 4ac}}{2a} \tag{14}$$

For right and left boundaries, the intersection points can be calculated geometrically,



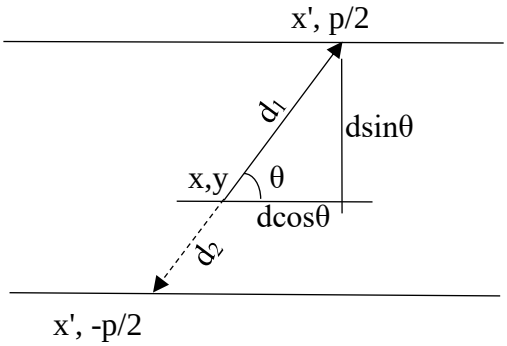
$$\pm\frac{p}{2} = x + d\cos\theta$$

Figure 4.7 Defining the vertical boundaries

The roots of this equation are d_1 and d_2.

$$d_{1,2} = \frac{\pm\frac{p}{2} - x}{\cos\theta} \tag{15}$$

For top and bottom boundaries, the intersection points can be calculated geometrically,



$$\pm\frac{p}{2} = y + d\sin\theta$$

Figure 4.8 Defining the horizontal boundaries

The roots of this equation are d_1 and d_2.

$$d_{1,2} = \frac{\pm\frac{p}{2} - y}{\sin\theta} \tag{16}$$

After comparing the interaction length with the smallest and positive distance, if interaction length is higher than smallest positive distance, then neutron will change the region. The next interaction length must be sampled according to neutronic properties of new region. Neutron is moved to closest surface by using the smallest distance, then x and y coordinates of next neutron location is calculated geometrically and stored. If interaction length is smaller than smallest positive distance, it means neutron is staying in same region and interacting. Neutron is moved to interaction point by using interaction length, then x and y coordinates of interaction location is calculated geometrically and stored.

4.5 SAMPLING INTERACTION TYPE

Once the neutron interaction length is compared to distances, if there is no region change, the neutron is moved, the interaction point and interaction type is sampled corresponding to cross sections. There are 3 different type of interactions that are sampled in QualifyingMC: fission, scattering and capture.

The cross section data ($\Sigma_f, \Sigma_c, \Sigma_s, \Sigma_t$) is chosen according to neutron energy and region material. The random numbers (ξ) decide interaction type as below,

$$\Sigma_t = \Sigma_f + \Sigma_c + \Sigma_s \tag{17}$$

$$0 < \xi < \frac{\Sigma_f}{\Sigma_t} \quad \rightarrow \quad Neutron\ undergoes\ fission$$

$$\frac{\Sigma_f}{\Sigma_t} < \xi < \frac{\Sigma_f + \Sigma_c}{\Sigma_t} \quad \rightarrow \quad Neutron\ undergoes\ capture$$

$$\frac{\Sigma_f + \Sigma_c}{\Sigma_t} < \xi < 1 \quad \rightarrow \quad Neutron\ undergoes\ scattering$$

Fission

If fission occurs 2 or 3 neutrons are born. The information is stored, and the neutron is killed and not being tracked anymore. The simulation continues for new neutron.

Capture

If capture occurs, the neutron is killed and not being tracked anymore. The simulation continues for new neutron

Scattering

If scattering occurs, neutron loses some energy in collision and the direction angle and the interaction length are sampled again. Neutron is being tracked.

Leakage

If neutron intersects with the boundary, it disappears at that boundary and counted as leaked, then appears at the opposite boundary with same direction angle as incoming neutron from neighbor pincell.

4.6 CALCULATION OF NEUTRON FLUX

The neutron flux is obtained with a parametrized neutron flux function following the usual three region neutron distribution. A Maxwell-Boltzmann [47] distribution is used to describe the distribution of neutrons in the thermal region.

For the fast neutron region, the flux is described by the Watt distribution [18]:

$$\Phi_{\mathrm{fs}}(E) = \mathrm{B}\cdot\exp\left(-\frac{E}{a}\right)\cdot\sinh(\sqrt{bE}) \tag{18}$$

Where,

$\Phi_{\mathrm{fs}}(E)$	B	$7.25(1)\times 10^{9}\ \mathrm{cm}^{-2}\,\mathrm{s}^{-1}$
	a	$1.21(1)\times 10^{6}\ \mathrm{eV}$
	b	$3.11(1)\times 10^{-6}\ \mathrm{eV}^{-1}$

The Watt distribution function is used to verify the calculated neutron flux distribution in the 2D reactor. The calculated neutron flux follows the flux pattern by Watt function, indicating that the QualifyingMC Monte Carlo method modeled neutron transport well. Figure 4.9 shows the comparison between these two estimates.

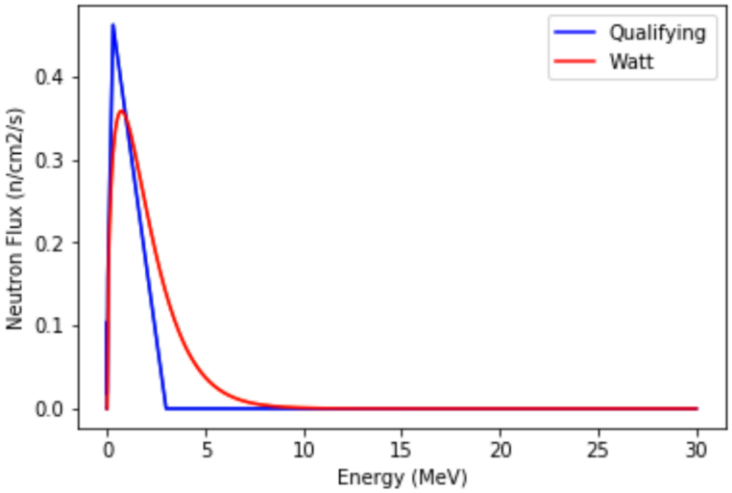


Figure 4.9 Neutron flux: QualifyingMC vs. Watt function

4.7 CALCULATION OF EFFECTIVE MULTIPLICATION FACTOR

The effective neutron multiplication factor, k_{eff}, is the average number of neutrons from one fission that cause another fission. The remaining neutrons either are absorbed in non-fission reactions or leave the system without being absorbed. The value of k_{eff} determines how a nuclear chain reaction proceeds. It is the ratio of the neutrons produced in one generation to neutrons lost in previous generation. Neutrons are only produced by fission reaction and lost by leakage and absorption. The remaining neutrons either are absorbed in non-fission reactions or leave the system without being absorbed. The value of k determines how a nuclear chain reaction proceeds.

4.8 HOW TO USE QualifyingMC

QualifyingMC is a python code, so any python compiler can be used to run the code. In Jupyter Notebook, the command QualifyingMC(N) is used, after compiling the

source code, to run the code for N number of neutrons. The following table shows the usage of command in Jupyter Notebook for N=10 Neutrons with results.

```
1  QualifyingMC(10)
```

```
Number of Neutrons.....................=  10
Number of Interactions.................=  133
Number of Scattering Events............=  128
Number of Capture Events...............=  5
Number of Fission Events...............=  5
Number of Absorption Events............=  10
Averega nu.............................=  2.2
Number of Neutrons Produced by Fission...=  11
Number of Neutrons Leaked from System....=  180
Number of Neutrons Leaked into System....=  180
Effective Multiplication Factor(keff)....=  1.0324324324324323
```

Table 4.1 QualifyingMC output for N (=10) neutrons

4.9 COMPARISON

In order to verify the computational methods and investigate the performance of QualifyingMC, the results are compared with reference results [16] obtained using same boundary conditions and taking into account the correlated physics. QualifyingMC approximates the neutron distribution and k_{eff} by estimating the number of fission neutrons produced per fission neutron started for a given generation. The ultimate goal of the code is to calculate k_{eff} and the flux distribution of neutrons to compare them with the reference results. Using QualifyingMC(N) command, the code is run for 10, 50, 100, 500, 1000 and 2000 neutrons. For each case, the code is run more than once. For example, for 10 neutrons, QualifyingMC(10) is run 5 times and the k_{eff} returned by each run is recorded. Then, their standard deviation, the dispersion of the k_{eff} values relative to its mean, is calculated as the square root of the variance. For each case i.e. 10, 50, 100, 500, 1000 and 2000 neutrons, the k_{eff} values and their standard deviation is calculated. Next, the values are compared with

the reference values, taking into account their sizes. The comparison is expressed as a ratio and is a unitless number. By multiplying these ratios by 100 these values are expressed as percentages. Furthermore, the relative difference is calculated by dividing the actual difference by the reference value. Assuming x the reference value and y the QualifyingMC value, the difference between the values, the actual difference, is calculated according to the following equation:

$$\Delta_{keff} = x - y \tag{19}$$

Thus, the absolute difference is,

$$|\Delta| = |x_{keff} - y| \tag{20}$$

The relative difference,

$$\text{Relative difference}(x, x_{\text{reference}}) = \frac{\text{Actual difference}}{x_{\text{reference}}} = \frac{\Delta}{x_{\text{reference}}} = \frac{x - x_{\text{reference}}}{x_{\text{reference}}} \tag{21}$$

CHAPTER 5

RESULTS AND DISCUSSION

QualifyingMC starts with a neutron point source inside the fuel cell for initiation of nuclear chain reaction. All calculations are run on HP 2.2 GHz Intel Core i5-5200 using up to 2000 source neutrons.

5.1 NEUTRON SOURCE AND CROSS SECTION

Neutrons are generated randomly using Numpy's "random" function and are propagated through space with certain energy and velocity. The U-235 nucleus interacts with the neutrons in two basic ways: it can scatter the interacting neutron deflecting it in a different direction while transferring some of its kinetic energy or it can capture the neutron, which in turn can affect the nucleus in several ways: absorption and fission. Scattering usually results in a change in the energy and direction of motion of a neutron but cannot directly cause the disappearance of a free neutron. The probability that the U-235 nucleus will scatter or capture a neutron is measured by its scattering cross section and capture cross section, respectively. The overall capture cross section is subdivided into other cross sections: the absorption cross section and the fission cross section. Absorption leads to the disappearance of free neutrons as a result of a nuclear reaction with fission or the formation of a new nucleus and another particle or particles such as protons, alpha particles and gamma ray photons. A neutron is captured when one or more neutrons collide and merge

with the U-235 to form a heavier nucleus i.e. U-236. Absorption cross section is often highly dependent on neutron energy. The likelihood of absorption is proportional to the time the neutron is in the vicinity of the nucleus. The time spent in the vicinity of the U-235 nucleus is inversely proportional to the relative velocity between the neutron and U-235 nucleus. Uncaptured neutrons undergo either absorption by fission or scattering. Each fission reaction results in two or three more neutrons that cause subsequent fission reaction(s). The fission cross section decreases monotonically with increasing energy of neutrons. If on average one neutron from each fission is captured and successfully produces fission, a self-sustaining chain reaction is produced. If on average more than one neutron from each fission triggers another fission, then the number of neutrons and the rate of energy production increases exponentially with time. When fast neutrons from the fission process collide with the reactor or shield material, they lose their energy to thermal equilibrium energy, where they have as much probability of gaining energy as losing it through further collisions.

5.2 NEUTRON TRANSPORT

QualifyingMC uses Monte Carlo methods to model neutron movement and interactions inside the reactor. The initial locations, direction angles and the distance a neutron can travel before undergoing interaction are chosen by random numbers. The diffusion of neutrons inside nuclear reactor resembles to the Brownian motion and can be analyzed in the stochastic framework as a random walk. Previous studies show that the use of random walk techniques for the analysis of neutron transport process in reactor is quite popular. The assumption is that the neutrons will be either absorbed or scattered at which point they

will change energy and direction. QualifyingMC has developed the neutron diffusion scenario for a two-dimensional case using the random walk framework. The neutrons are distributed randomly in fuel region moving through the fissile content uranium dioxide fuel, aluminum cladding material and water moderator.

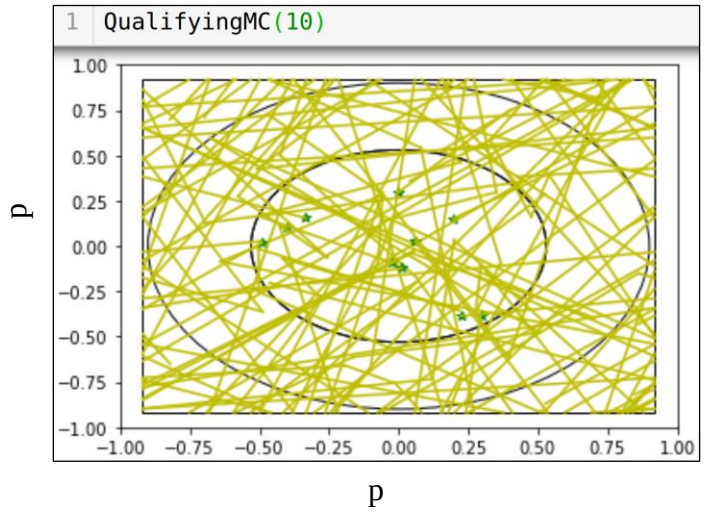


Figure 5.1 Visual presentation of neutron transport by QualifyingMC for 10 neutrons

Table 5.1 QualifyingMC calculates k_{eff} for 10 neutrons

```
1 QualifyingMC(10)

Number of Neutrons.....................=  10
Number of Interactions.................=  133
Number of Scattering Events............=  128
Number of Capture Events...............=  5
Number of Fission Events...............=  5
Number of Absorption Events............=  10
Averega nu.............................=  2.2
Number of Neutrons Produced by Fission...=  11
Number of Neutrons Leaked from System....=  180
Number of Neutrons Leaked into System....=  180
Effective Multiplication Factor(keff)....=  1.0324324324324323
```

As discussed above, the energy dependent cross sections of these materials are used for calculations and periodic boundary condition is applied to pincell boundaries. Thus, neutrons are able to leave the reactor without undergoing interaction and appear from the

other side of the reactor with the same energy and direction. The study of the motions and interactions of neutrons with materials is important because it gives information about where neutrons are in an apparatus, what direction they are going, and how quickly they are moving. This also helps to determine the behavior of nuclear reactor cores and experimental or industrial neutron beams.

5.3 EFFECTIVE NEUTRON MULTIPLICATION FACTOR

The effective neutron multiplication factor calculated with Monte Carlo code QualifyingMC using detailed geometrical model of the reactor was compared to the reference values. Table 5.2 reports the k_{eff} calculated with QualifyingMC and the reference values. The values are in reasonably good agreement indicating that the material and geometrical properties of the reactor core are modelled well. For higher numbers of source neutrons, the QualifyingMC k_{eff} values are closer to unity (Table 5.2). Table 5.3 shows the comparison between k_{eff} values calculated by QualifyingMC and the reference values according equation 21. A negative relative difference indicates that the QualifyingMC value is greater than the reference value. A positive relative difference, on the other hand, means, the QualifyingMC value is smaller compared to the reference value. The results are shown in Figure 5.2.

Table 5.2 Effective multiplication factors calculated with different models

Code	**Library**	k_{eff}	**CPU time (min)**
HEXNOD	ENDF/B-IV	0.99811	N/A
MCU	MCUDAT	0.99713	N/A
HELIOS	ENDF/B-VI	1.00090	N/A
BOXER	JEF-1	1.00094	N/A
PSG	ENDF/B-VI.8	0.99516 ± 0.00022	40.69
MORA 2-group	ENDF/B-VI.8	1.04296 ± 0.00027	14.27
MORA 4-group	ENDF/B-VI.8	1.01417 ± 0.00027	14.28
Mora 69-group	ENDF/B-VI.8	1.00099 ± 0.00028	15.97
Mora 172-group	ENDF/B-VI.8	1.00077 ± 0.00021	16.50
QualifyingMC (10)	ENDF/B-VIII	1.12621 ± 0.0343	0.03
QualifyingMC (50)	ENDF/B-VIII	1.09298 ± 0.0340	0.5
QualifyingMC (100)	ENDF/B-VIII	1.06071 ± 0.0322	2
QualifyingMC (500)	ENDF/B-VIII	1.03243 ± 0.0312	8
QualifyingMC (1000)	ENDF/B-VIII	1.01524 ± 0.0027	30
QualifyingMC (2000)	ENDF/B-VIII	1.00745 ± 0.0021	240

Table 5.3 Comparison of the effective multiplication factors.

Reference – QualifyingMC (QMC)	Relative difference
PSG - QMC	-0.01229
MORA 2 group - QMC	0.03524
MORA 4 group - QMC	0.00667
MORA 69 group - QMC	-0.00641
MORA 172 group - QMC	-0.00663
HEXNOD - QMC	-0.00927
MCU - QMC	-0.01024
HELIOS - QMC	-0.00650
BOXER - QMC	-0.00646

As shown in Table 5.2 and Figure 5.3, the results are in reasonably good agreement (99.8%) with some variation. In part, this results from the fact that the list of precursor isotopes included in each group depends on the cross-section library used in the calculations [8]. Also, the references use 20,000 source neutrons in their calculation while QualifyingMC uses up to 2,000 neutrons due to resource limitations. When QualifyingMC uses higher numbers of source neutrons the results are in relatively better agreement: 87.3% (N=10) vs. 99.8% (N=2000).

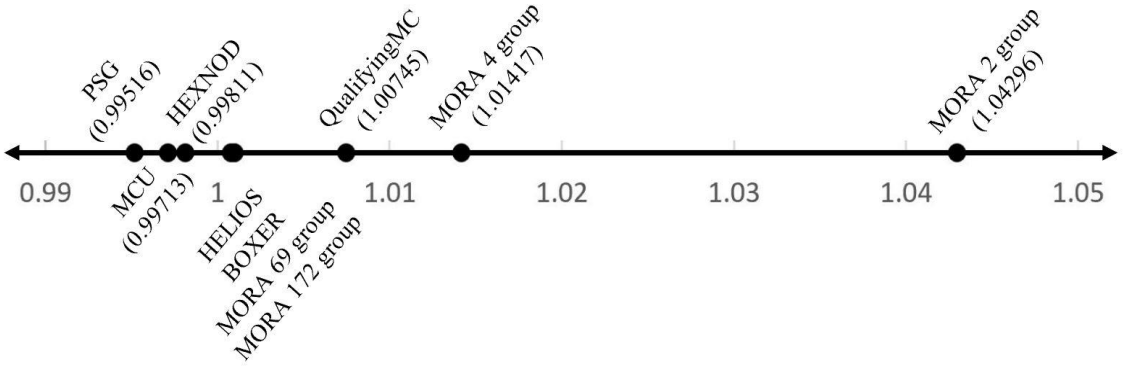


Figure 5.2 Comparison of QualifyingMC to reference effective multiplication factors

Furthermore, increasing the energy resolution will result in a more accurate description of the resonance region, which will result in further better agreement. Figure 5.3 shows the relation between k_{eff} and number of neutrons used by QualifyingMC. When the Monte Carlo method is run for higher number of neutrons a good approximation of the multiplication factor is achieved.

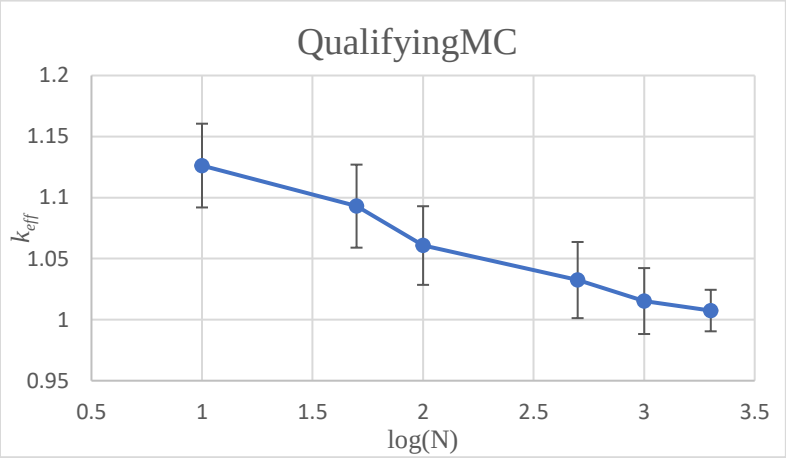


Figure 5.3 QualifyingMC k_{eff} approaches unity when more neutrons are used

CHAPTER 6

SUMMARY AND CONCLUSION

Monte Carlo simulations are widely used for modeling nuclear reactor operations. Although Monte Carlo calculations are costly, they give a result of exactly known accuracy and the calculation costs can be adjusted to meet the required precision. In this study, Monte Carlo method was used to model the motion and interactions of neutrons inside a 2D reactor and calculate the neutron flux distribution and effective multiplication factors inside the reactor. The initial locations, direction angles and the neutron mean free path are defined using random numbers. Freely moving neutrons inside reactor are either absorbed or scattered. Some neutrons that are absorbed underwent fission. Scattered neutrons changed energy and direction. Neutrons leaving reactor without undergoing interaction appeared from the other side of the reactor with the same energy and direction. This process continued until the neutron is absorbed or further scattered. Ultimately, the neutron flux distribution and effective multiplication factors were calculated, and results were compared with reference results.

Despite the widespread use of periodic boundary conditions in computer simulations, there are some limitations associated with the choice. The physical interpretation and consequences of the artificial infinite periodicity is unclear. The symmetry of the reactor imposed by the periodic replicas on the whole system can be problematic. Nonetheless, the

periodic boundary condition has a number of positive features, and in general circumstances preferable to the solid boundary conditions. The energy dependent cross section used in this study was obtained from the ENDF library and divided into 13 equal energy groups. Based on the comparison between multigroup and continuous energy Monte Carlo calculations, the multigroup energy Monte Carlo results are within 10% of the continuous energy Monte Carlo results. This calculational methodology, however, may contain uncertainties caused by several factors such as the multigroup library, multi-dimensionality effects, the "ray-effect", and geometric approximations, though previous studies have shown that the effect of geometric approximations is negligible and the use of a S_8 quadrature set is adequate for these calculations i.e. the "ray-effect" is negligible. However, a better approximation can be made by three-dimensional full-core calculation using continuous energy cross section.

QualifyingMC was compared with reference codes that use 20,000 source neutrons in their calculation. QualifyingMC, however, uses up to 2,000 neutrons. The calculations are run on HP 2.2 GHz Intel Core i5-5200, which takes 240 min CPU time to complete the calculation for 2,000 source neutrons. A time-dependent full-core calculation using continuous energy cross section and higher number of source neutrons, though would be computationally expensive, is likely to give a better approximation. However, within the scope of this study, the results were shown to be in reasonably good agreement, proving that Monte Carlo method is a useful technique to simulate neutron kinetics based on repeated random sampling and statistical analysis, and capable of modeling engineering problems relating to probabilistic interpretation.

REFERENCES

[1] Lewis EE, Miller WF Jr (1993) Computational methods of neutron transport. American Nuclear Society, La Grange Park, Illinois

[2] Wilson RPH, Feder R, Fischer U et al (2008) State-of-art 3-D radiation transport methods for fusion energy system. Fusion Eng Des 83:824–833

[3] Wu YC, Song J, Zheng HQ et al (2015) CAD-Based Monte Carlo Program for integrated simulation of nuclear system SuperMC. Ann Nucl Energy 82:161–168),

[4] Nigel SMITH. Malcolm ARMISHAW. Andrew COOPER, "Current Status and Future Direction of the MONK Software Package," Proceedings of an International Conference on Nuclear Criticality Safety, ICNC'03, Tokai-mura, Japan, Oct. 20-24 (2003).

[5] Joachim Miss, Olivier Jacquet, Franck Bernard, Isabelle Duhamel, St ephane Evo, Marc Leclere, "Using various point wise and multi-group cross section libraries in MORET criticality calculations," International Conference on Nuclear Data for Science and Technology 2007, Nice, France, April 22-27 (2007)

[6] Nagaya Yasunobu, Okumura Keisuke, Mori Takamasa, Nakagawa Nasayuki, "mvp/gmvp ii: General Purpose Monte Carlo Codes for Neutron and Photon Transport Calculations based on Continuous Energy and Multigroup Methods," Nippon Genshiryoku Kenkyujo JAERI Repoto, p.409 (2005)

[7] Jaakko Leppänen, “A New Assembly-level Monte Carlo Neutron Transport Code for Reactor Physics Calculations,” Mathematics and Computation, Supercomputing, Reactor Physics and Nuclear and Biological Applications, Palais des Papes, Avignon, France, September 12-15 (2005)

[8] X-5 Monte Carlo Team, “MCNP-A General Monte Carlo N-particle Transport Code Version 5,” Los Alamos National Lab, LA-CP-03-0248 (2003).

[9] R. N. Blomquist, “Status of the VIM Monte Carlo Neutron/Photon Transport Code,” ANS 12th Biennial RPSD Topical Meeting, Santa Fe, NM, April 14-18 (2002).

[10] S.M. Bowman, D.F. Hollenbach, M.D. DeHart, B.T. Rearden, I.C. Gauld, and S. Goluoglu, “SCALE 5: Powerful New Criticality Safety Analysis Tools,” Proc. Of the 7th International Conference on Nuclear Criticality Safety (ICNC2003), Tokai-mura, Japan, October 20-24 (2003).

[11] R. Procassini, J. Taylor, S. Hagmann, “Update on the Development and Validation of MERCURY: a Modern, Monte Carlo Particle Transport Code,” Lawrence Livermore National Lab, UCRL-PROC-212722 (2005).

[12] T.M. Sutton, T.J. Donovan et al, “The MC21 Monte Carlo Transport Code,” Joint International Topical Meeting on Mathematics & Computation and Supercomputing in Nuclear Applications (M&C + SNA 2007), Monterey, CA, April 15-19 (2007).

[13] Wagner, John C.; Peplow, Douglas E.; Mosher, Scott W.: FW-CADIS Method for Global and Reginal Variance Reduction of Monte Carlo Radiation Transport calculations. In: Nuclear Science and Engineering 176 (2014).

[14] J.C. Wagner, A. Haghighat, and B.G. Petrovic, Multigroup versus continuous energy MCNP for PWR fluence calculations. Nuclear Engineering Department. The Pennsylvania State University.

[15] B.G. PETROVIC and A. HAGHIGHAT, "Effect of the Quadrature Order on the Accuracy of Fluence Calculations," *Trans. Am. Nucl. Soc.*, 68, 477-479 (June 1993).

[16] Leppänen, Jaakko (2008), On the Calculation of Reactor Time Constants Using the Monte Carlo Method, IYNC 2008, Interlaken, Switzerland, Paper No. 116.

[17] Parette J.M. et al., "A benchmark on the calculation of kinetic parameters based on reactivity effect experiments in the CROCUS reactor", Annals of Nuclear Energy, Vol 33, Issue 8, May 2006. pp. 739-748.

[18] Watt, B.E., 1952. Energy spectrum of neutrons from thermal fission of U235. Phys. Rev. 87 (6), 1037–1041. http://dx.doi.org/10.1103/PhysRev.87.1037.

[19] Gacuci, Dan G.: Handbook of Nuclear Engineering. Springer, 2010.

[20] Larsen, E. ; Morel, J. E.: Advances in Discrete Ordinates Methodology. Springer, Berlin, 2009.

[21] Buffon, G. "Essai d'arithmétique morale." *Histoire naturelle, générale er particulière, Supplément* **4**, 46-123, 1777.

[22] (Metropolis 1987).

[23] (Kroese, D. P.; Brereton, T.; Taimre, T.; Botev, Z. I. (2014). "Why the Monte Carlo method is so important today". WIREs Comput Stat. 6 (6): 386–392. doi:10.1002/wics.1314.)

[24] Hubbard, Douglas; Samuelson, Douglas A. (October 2009). "Modeling Without Measurements". OR/MS: 28–33.

[25] Haghighat, A.; Wagner, John C.: Monte Carlo Variance Reduction with Deterministic Importance Functions. In: Progress in Nuclear Energy 42 (2003), S. 25.

[26] Peplow, Douglas E.: Comparison of HybridMethods for Global Variance Reduction in Shielding Calculations. In: Transactions of the American Nuclear Society 107 (2012).

[27] Davis, Andrew; Turner, Andrew: Improving Computational Efficiency Of Monte-Carlo Simulations with Variance Reduction. In: International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (2013).

[28] Wagner, John C.; Peplow, Douglas E. ; Mosher, ScottW. ; Evans, Thomas M.: Review of Hybrid (Deterministic/Monte Carlo) Radiation Transport Methods, Codes and Applications at Oak Ridge National Laboratory. In: Nuclear Science and Technology 2 (2011), S. 808–814.

[29] Wagner, John C.; Peplow, Douglas E.; Mosher, Scott W.: FW-CADIS Method for Global and Reginal Variance Reduction of Monte Carlo Radiation Transport calculations. In: Nuclear Science and Engineering 176 (2014).

[30] Cashwell, E.D.; Everett, C.J. (1959). A Practical Manual on the Monte Carlo Method for Random Walk Problems (PDF). London: Pergamon Press.

[31] L.M. Petrie and N.F. Landers. An improved monte carlo criticality program. NUREG/CR-0200, 2, 1981.

[32] Jaakko Lepp¨anen. Serpent - a continuous-energy Monte Carlo reactor physics burnup calculation code. VTT Technical Research Centre of Finland, 4, 2013.

[33] T. E. Booth, “Adaptive Importance Sampling with a Rapidly Varying Importance Functional Los Alamos National Laboratory document, LA-UR-99-3611 (1999).

[34] Leppänen, Jaakko; Pusa, Maria; Viitanen, Tuomas; Valtavirta, Ville; Kaltiaisenaho, Toni (2016). The Serpent Monte Carlo code: Status, development and applications in 2013. Annals of Nuclear Energy. 82: 142–150. doi:10.1016/j.anucene.2014.08.024

[35] Leppänen, Jaakko. "Serpent - a Continuous-energy Monte Carlo Reactor Physics Burnup Calculation Code User's Manual" (PDF). Retrieved 4 November 2018.

[36] Cashwell, E.D.; Everett, C.J. (1959). A Practical Manual on the Monte Carlo Method for Random Walk Problems (PDF). London: Pergamon Press.

[37] Leppänen, Jaakko. "Greetings from the Serpent developer team & current status and future plans for Serpent 2" (PDF). Retrieved 11 November 2018.

[38] Leppänen, Jaakko; Pusa, Maria; Fridman, Emil (2016). "Overview of methodology for spatial homogenization in the Serpent 2 Monte Carlo code". Annals of Nuclear Energy. 96: 126–136. doi:10.1016/j.anucene.2016.06.007.

[39] Travis E, Oliphant. A guide to NumPy, USA: Trelgol Publishing, (2006).

[40] Makoto Matsumoto and Takuji Nishimura. 1998. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput.Simul.*8, 1 (January 1998), 3-30. DOI: https://doi.org/10.1145/272991.272995.

[41] Faezeh Abbasiaus Teheran, Iran (2013). Monte Carlo Based Modeling and Simulation off Neutron Flux Distribution and Activity Map of the German Research Reactor FRJ-2. Diese Dissertation ist auf den Internetseiten der Hochschulbibliothek online verfügbar.

[42] Simon, H. (1955) On a Class of Skew Distribution Functions. Biometrika: 42(3/4).

[43] ShimPlotWell. http://www.cross-section-plotter.com.

[45] Nuclear Power for Everybody. nuclear-power.net.

[46] https://slideplayer.com/slide/10946413/

[47] Westcott, C., 1955. The specification of neutron flux and nuclear cross-sections in reactor calculations. J. Nucl. Energy 2, 59–76. http://dx.doi.org/10.1016/0891-3919(55) 90017-6.

[49] Y. Oka (ed.), Nuclear Reactor Design, An Advanced Course in Nuclear Engineering 2, DOI 10.1007/978-4-431-54898-0_2, ©Authors 2014

APPENDIX A: QualifyingMC code

```
############################################################
#          2D MONTE CARLO NEUTRON TRANSPORT CODE           #
############################################################
import random
import math
import pylab
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import maxwell
from scipy.interpolate import interp1d
np.random.seed()

def CrossSections(Energy,region):

#============================================================#
#     Energy Dependent Macroscopic Cross Section Data        #
#============================================================#

    #Unit is 1/cm
    #sigma_x[EnergyGroup][Region]
    sigma_f=np.array([[1.05e-1,0,0],
                      [5.96e-2,0,0],
                      [6.02e-2,0,0],
                      [1.06e-1,0,0],
                      [2.46e-1,0,0],
                      [2.50e-1,0,0],
                      [1.07e-1,0,0],
                      [1.28e+0,0,0],
                      [9.30e+0,0,0],
                      [2.58e+1,0,0]])
    sigma_c=np.array([[1.41e-6,1.71e-2,3.34e-6],
                      [1.34e-3,7.83e-3,3.34e-6],
                      [1.10e-2,2.83e-4,2.56e-7],
                      [3.29e-2,4.52e-6,6.63e-7],
                      [8.23e-2,1.06e-5,2.24e-7],
                      [4.28e-2,4.39e-6,1.27e-7],
                      [9.90e-2,1.25e-5,2.02e-7],
                      [2.51e-1,3.98e-5,6.02e-7],
                      [2.12e+0,1.26e-4,1.84e-6],
```

```
                    [4.30e+0,3.95e-4,5.76e-6]])
    sigma_s=np.array([[2.76e-1,1.44e-1,1.27e-2],
                      [3.88e-1,1.76e-1,7.36e-2],
                      [4.77e-1,3.44e-1,2.65e-1],
                      [6.88e-1,2.66e-1,5.72e-1],
                      [9.38e-1,2.06e-1,6.69e-1],
                      [1.52e+0,2.14e-1,6.81e-1],
                      [2.30e+0,2.23e-1,6.82e-1],
                      [2.45e+0,2.31e-1,6.83e-1],
                      [9.79e+0,2.40e-1,6.86e-1],
                      [4.36e+1,2.41e-1,6.91e-1]])
    sigma_t=sigma_f+sigma_c+sigma_s


#============================================================#
#                         Energy Group                       #
#============================================================#

    #Unit is MeV
    Group_Energy=[3e+1,    #Group0
                  3e+0,    #Group1
                  3e-1,    #Group2
                  3e-2,    #Group3
                  3e-3,    #Group4
                  3e-4,    #Group5
                  3e-5,    #Group6
                  3e-6,    #Group7
                  3e-7,    #Group8
                  3e-8]    #Group9
    for g in range(len(Group_Energy)):
        if Energy>=Group_Energy[g]:
            #Neutron in Group g
            group=g
            break
    #print(group,Energy)


#============================================================#
#                         Cross Sections                     #
#============================================================#

    #Unit is 1/cm
    sig_f=sigma_f[group][region]
    sig_c=sigma_c[group][region]
    sig_s=sigma_s[group][region]
    sig_t=sigma_t[group][region]
    sig=[sig_f,sig_c,sig_s,sig_t]
```

```
    return sig

def CalcGroup(Energy):
    #Unit is MeV
    Group_Energy=[3e+1,    #Group0
              3e+0,    #Group1
              3e-1,    #Group2
              3e-2,    #Group3
              3e-3,    #Group4
              3e-4,    #Group5
              3e-5,    #Group6
              3e-6,    #Group7
              3e-7,    #Group8
              3e-8]    #Group9
    for g in range(len(Group_Energy)):
        if Energy>=Group_Energy[g]:
            #Neutron in Group g
            group=g
            break
    return group

def QualifyingMC(Neutrons_Number):

#============================================================#
#                        Initialization                      #
#============================================================#

print("Number  of  Neutrons.....................=  ",Neu-
trons_Number)  # Number of initial neutrons
    Neutrons_Produced=0                                     #
Number of fission neutrons
    interaction_point_x=[]                                  #
list for x coordinate of interaction point
    interaction_point_y=[]                                  #
list for y coordinate of interaction point
    FuelSurfNeuNum=[0,0,0,0,0,0,0,0,0,0]
    CladSurfNeuNum=[0,0,0,0,0,0,0,0,0,0]
    Group_Energy=[3e+1,    #Group0
              3e+0,    #Group1
              3e-1,    #Group2
              3e-2,    #Group3
              3e-3,    #Group4
              3e-4,    #Group5
              3e-5,    #Group6
              3e-6,    #Group7
              3e-7,    #Group8
              3e-8]    #Group9
```

```
    Fission=0                              # Number of fission
interactions
    nu=0
    Capture=0                             # Number of absorption
interactions
    Absorption=0                          # Number of absorption
interactions
    Scattering=0                         # Number of scattering
interactions
    Leakage=0                              #  Number  of  leaked
neutrons


#============================================================#
#                          Geometry                          #
#============================================================#

r_fuel     = 0.53                                        # Fuel
radius
    r_clad_in  = 0.53                                        #
Cladding inner radius
    r_clad_out = 0.90                                        #
Cladding outer radius
    pitch      = 1.837                                       #
Cell pitch
    t_clad     = r_clad_out-r_clad_in                        #
Cladding thickness


#============================================================#
#        Spatial and Energy Distribution of Neutrons         #
#============================================================#

Neutrons_Energy=maxwell.rvs(size=Neutrons_Number)  # Maxwel-
lian Neutron Energy Distribution [MeV]
    theta1=2*np.pi*np.random.random((Neutrons_Number)) # An-
gular location of initial neutron over 2pi
    rnd=np.random.power(2,size=(Neutrons_Number))     # Ran-
dom number for radial locations(powerlaw distribution)
    r=rnd*r_fuel                                             #
Radial locations of initial neutrons over r


#============================================================#
#                       Calculations                         #
#============================================================#
```

```
for i in range(Neutrons_Number):                    # Loop for
each neutron
    #     print("Neutron ",i)
        # Initialize parameters
        #----------------------
        alive=1                                              #
Neutron life parameter
        interaction=0                                        #
Interaction control parameter
        boundary=0                                           #
Boundary control parameter
        regionchange=0                                       #
Region change control parameter
        surface=0                                            #
Surface control parameter
        region=0                                             #
Region control parameter, [fuel,cladding,moderator]=[0,1,2]
        # Initilize neutron location and energy
        #--------------------------------------
        E = []
        E.append(Neutrons_Energy[i])
        Energy=E[-1]
        x = []                                               #
list for x coordinate of neutron location
        y = []                                               #
list for y coordinate of neutron location
        x.append(r[i]*np.cos(theta1[i]))                     #
Initial x coordinate of radial neutron location
        y.append(r[i]*np.sin(theta1[i]))                     #
Initial y coordinate of radial neutron location
        plt.plot(x,y,'*g')                                   #
Mark initial position of neutron

        # Track neutron while it is alive
        #--------------------------------
        while alive==1:
            sig=CrossSections(Energy,region)                 #
Load Cross Sections sig=[sig_f,sig_c,sig_s,sig_t]

#=========================================================#
#                       Fuel Region                       #
#=========================================================#

if region==0:
    #             print("fuel(UO2)")
                # UO2 Fuel
```

```
                A=238.02891                                     #
Mass Number(A) of Uranium
                density=10.97                                   #
g/cc
                if regionchange==0:
    #                 print("new theta")
                    theta=2*np.pi*np.random.random()            #
sample new direction angle
                else:
    #                 print("same theta")
                    theta=theta                                 #
keep direction angle same
                    regionchange=0
                d=-(1/sig[3])*np.log(np.random.random())#
the distance neutron goes before interaction
    #             print(d,theta)
                # Check distance for nearest surface
                #---------------------------------------
--
                a=1
                b=2*(x[-1]*np.cos(theta)+y[-
1]*np.sin(theta))
                c=x[-1]**2+y[-1]**2-r_fuel**2
                delta=b**2-4*a*c

                df = []
                d_pos = []
                if delta>=0:
                    df1=(-b-delta**0.5)/(2*a)
                    df.append(df1)
                    df2=(-b+delta**0.5)/(2*a)
                    df.append(df2)
                else:
    #                 print("delta cannot be negative")
                    break
                for k in range(len(df)):
                    if df[k]>1e-9:
                        d_pos.append(df[k])
                d_pos.sort()
    #             print(d)
    #             print(df)
                dmin=d_pos[0]
    #             print(dmin)

                if d>=dmin:
                    # Neutron is leaving the fuel region(0)
and entering the cladding region(1)
```

```
    #                 print("fuel ->  cladding")
                    FuelSurfNeuNum[CalcGroup(Energy)]      =
FuelSurfNeuNum[CalcGroup(Energy)]+1
                    regionchange=1
                    surface=1
                    region=1
                    x.append(x[-1]+dmin*np.cos(theta))
                    y.append(y[-1]+dmin*np.sin(theta))
                    plt.plot(x[-2:],y[-2:],'-y')
                else:
                    # Neutron is interacting at fuel region
    #                 print("interaction in fuel")
                    interaction=1
                    x.append(x[-1]+d*np.cos(theta))      # x
coordinate of projected interaction point
                    y.append(y[-1]+d*np.sin(theta))      # y
coordinate of projected interaction point
                    plt.plot(x[-2:],y[-2:],'-y')


#=========================================================#
#                       Cladding Region                   #
#=========================================================#

if region==1:
                # Aluminum Cladding
                A=26.981539                                 #
Mass Number(A) of Zirconium
                density=2.70                                #
g/cc
    #             print("cladding(Zirconium)")
                if regionchange==0:
    #                 print("new theta")
                    theta=2*np.pi*np.random.random()        #
sample new direction angle
                else:
    #                 print("same theta")
                    theta=theta                             #
keep direction angle same
                    regionchange=0
                d=-(1/sig[3])*np.log(np.random.random())#
the distance neutron goes before interaction
    #             print(d,theta)

                # Check distance for nearest surface
                #----------------------------------------
--
```

```
                a=1
                b=2*(x[-1]*np.cos(theta)+y[-
1]*np.sin(theta))
                c_in=x[-1]**2+y[-1]**2-r_clad_in**2
                c_out=x[-1]**2+y[-1]**2-r_clad_out**2
                delta_in=b**2-4*a*c_in
                delta_out=b**2-4*a*c_out
                dc = []
                d_pos = []
                #=
                if delta_in>=0:
                    dc1=(-b-delta_in**0.5)/(2*a)
                    dc.append(dc1)
                    dc2=(-b+delta_in**0.5)/(2*a)
                    dc.append(dc2)
                    if delta_out>=0:
                        dc3=(-b-delta_out**0.5)/(2*a)
                        dc.append(dc3)
                        dc4=(-b+delta_out**0.5)/(2*a)
                        dc.append(dc4)
                    else:
    #                   print("delta_out cannot be negative")
                        break
                else:
    #                   print("no intersection with cladding
inner surface")
                    dc1=1e100
                    dc2=1e100
                    if delta_out>=0:
                        dc3=(-b-delta_out**0.5)/(2*a)
                        dc.append(dc3)
                        dc4=(-b+delta_out**0.5)/(2*a)
                        dc.append(dc4)
                    else:
    #                   print("delta_out cannot be negative")
                        break
                for k in range(len(dc)):
                    if dc[k]>1e-9:
                        d_pos.append(dc[k])
                d_pos.sort()
    #             print(d)
    #             print(dc)
                dmin=d_pos[0]
    #             print(dmin)

                if d>=dmin:
```

```
                    # Neutron is leaving the cladding re-
gion(1)
                    CladSurfNeuNum[CalcGroup(Energy)]      =
CladSurfNeuNum[CalcGroup(Energy)]+1
                    regionchange=1
                    if dmin==dc1 or dmin==dc2:
    #                     print("cladding ->  fuel")
                        region=0
                        surface=1
                    elif dmin==dc3 or dmin==dc4:
    #                     print("cladding ->  moderator")
                        region=2
                        surface=1
                    x.append(x[-1]+dmin*np.cos(theta))
                    y.append(y[-1]+dmin*np.sin(theta))
                    plt.plot(x[-2:],y[-2:],'-y')
                else:
                    # Neutron is interacting at cladding re-
gion
    #                 print("interaction in cladding")
                    interaction=1
                    x.append(x[-1]+d*np.cos(theta))      # x
coordinate of projected interaction point
                    y.append(y[-1]+d*np.sin(theta))      # y
coordinate of projected interaction point
                    plt.plot(x[-2:],y[-2:],'-y')


#=========================================================#
#                     Moderator Region                    #
#=========================================================#

if region==2:
                # H2O Moderator
                A=1.00794                                  #
Mass Number(A) of H
                density=1                                  #
g/cc
    #             print("moderator(water)")
                if regionchange==0:
    #                 print("new theta")
                    theta=2*np.pi*np.random.random()       #
sample new direction angle
                else:
    #                 print("same theta")
                    theta=theta                            #
keep direction angle same
```

```
                    regionchange=0
                d=-(1/sig[3])*np.log(np.random.random())#
the distance neutron goes before interaction
    #             print(d,theta)

                # Check distance for nearest surface
                #-----------------------------------------
--
                a=1
                b=2*(x[-1]*np.cos(theta)+y[-
1]*np.sin(theta))
                c_out=x[-1]**2+y[-1]**2-r_clad_out**2
                delta_out=b**2-4*a*c_out

                dm = []
                d_pos = []
                if delta_out>=0:
                    dm1=(-b-delta_out**0.5)/(2*a)
                    dm.append(dm1)
                    dm2=(-b+delta_out**0.5)/(2*a)
                    dm.append(dm2)
                    dm3=(pitch/2-x[-1])/np.cos(theta)      #
distance to right boundary
                    dm.append(dm3)
                    dm4=(pitch/2-y[-1])/np.sin(theta)      #
distance to top boundary
                    dm.append(dm4)
                    dm5=(-pitch/2-x[-1])/np.cos(theta)     #
distance to left boundary
                    dm.append(dm5)
                    dm6=(-pitch/2-y[-1])/np.sin(theta)     #
distance to bottom boundary
                    dm.append(dm6)
                else:
    #                   print("no intersection with cladding
outer surface")
                    dm1=1e100
                    dm2=1e100
                    dm3=(pitch/2-x[-1])/np.cos(theta)      #
distance to right boundary
                    dm.append(dm3)
                    dm4=(pitch/2-y[-1])/np.sin(theta)      #
distance to top boundary
                    dm.append(dm4)
                    dm5=(-pitch/2-x[-1])/np.cos(theta)     #
distance to left boundary
                    dm.append(dm5)
```

```
                    dm6=(-pitch/2-y[-1])/np.sin(theta)     #
distance to bottom boundary
                    dm.append(dm6)

                for k in range(len(dm)):
                    if dm[k]>1e-9:
                        d_pos.append(dm[k])
                d_pos.sort()
    #             print(d)
    #             print(dm)
                dmin=d_pos[0]
    #             print(dmin)

                if d>=dmin:
                    # Neutron is leaving the moderator re-
gion(2)
                    regionchange=1
                    x.append(x[-1]+dmin*np.cos(theta))
                    y.append(y[-1]+dmin*np.sin(theta))
                    plt.plot(x[-2:],y[-2:],'-y')
                    if dmin==dm1 or dmin==dm2:
    #                     print("moderator ->  cladding")
                        region=1
                    elif dmin==dm3:
    #                     print("leaving from right boundary")
                        boundary=1
                        Leakage=Leakage+1
    #                     print("appeared from left boundary")
                        x.append(-x[-1])
                        y.append(y[-1])
                    elif dmin==dm4:
    #                     print("leaving from top boundary")
                        boundary=1
                        Leakage=Leakage+1
    #                     print("appeared at bottom boundary")
                        x.append(x[-1])
                        y.append(-y[-1])
                    elif dmin==dm5:
    #                     print("leaving from left boundary")
                        boundary=1
                        Leakage=Leakage+1
    #                     print("appeared at right boundary")
                        x.append(-x[-1])
                        y.append(y[-1])
                    elif dmin==dm6:
    #                    print("leaving from bottom boundary")
                        boundary=1
```

```
                        Leakage=Leakage+1
    #                     print("appeared at top boundary")
                        x.append(x[-1])
                        y.append(-y[-1])
                else:
                    # Neutron is interacting at moderator re-
gion
    #                 print("interaction in cladding")
                    interaction=1
                    x.append(x[-1]+d*np.cos(theta))      # x
coordinate of projected interaction point
                    y.append(y[-1]+d*np.sin(theta))      # y
coordinate of projected interaction point
#                     plt.plot(x[-2:],y[-2:],'-y')


#============================================================#
#                          Interactions                      #
#============================================================#

if interaction==1:
                #sig=[sig_f,sig_c,sig_s,sig_t]
                interaction=0
                interaction_point_x.append(x[-1])
                interaction_point_y.append(y[-1])
                # Find interaction type
                rnd=np.random.random()
                if rnd<=(sig[0]/sig[3]):
    #                 print("Fisson")
                    Fission=Fission+1
                    alive=0
                    if np.random.random()<0.5:
                        nf=2
                    else:
                        nf=3
                    Neutrons_Produced=Neutrons_Produced+nf
                    nu=Neutrons_Produced/Fission
                elif         (sig[0]/sig[3])<rnd         and
rnd<=((sig[0]+sig[1])/sig[3]):
    #                 print("Capture")
                    Capture=Capture+1
                    alive=0
                else:
    #                 print("Scattering")
                    Scattering=Scattering+1
                    #====================
                    # Neutron Slowing Down
```

```
                    #====================
                    ksi=1+np.log((A-1)/(A+1))*(A-
1)**2/(2*A) # Collision Energy Loss Parameter
                    E.append(E[-1]*np.exp(-ksi))           #
Average Energy of Neutron after collision
                    Energy=E[-1]
                    #plt.plot(interaction_point_x,interac-
tion_point_y,'*r')


#============================================================#
#                     Plotting Geometry                      #
#============================================================#

plt.gcf().gca().add_artist(plt.Cir-
cle((0,0),r_fuel,fill=False))
    plt.gcf().gca().add_artist(plt.Cir-
cle((0,0),r_clad_in,fill=False))
    plt.gcf().gca().add_artist(plt.Cir-
cle((0,0),r_clad_out,fill=False))
    plt.gcf().gca().add_artist(plt.Rectangle((-pitch/2,-
pitch/2),width=pitch, height=pitch, fill=False))
    plt.xlim(-1,1)
    plt.ylim(-1,1)

    # Neutron Flux Spectrum
    A = np.pi*r_fuel**2
    c = Neutrons_Number*10 #scaling factor
    x1=Group_Energy
    y1=np.array(FuelSurfNeuNum)/np.array(A*c)
    x2 = np.linspace(0,30,500)
    y2 = 0.453 * np.sinh((2.29*x2)**0.5) * np.exp(-x2*1.036)
    plt.figure()
    plt.plot(x1,y1, x2,y2)
    plt.xlabel("Energy (MeV)")
    plt.ylabel("Flux (n/cm2/s)")
    pylab.plot(x1, y1, '-b', label='Qualifying')
    pylab.plot(x2, y2, '-r', label='Watt')
    pylab.legend(loc='upper right')


#============================================================#
#                           Results                          #
#============================================================#

Neutrons_Lost=Leakage+Absorption+Fission
    keff=(Neutrons_Produced+Leakage)/Neutrons_Lost
```

```
    Interactions=Scattering+Absorption+Fission
    Absorption=Fission+Capture
    print("Number of Interactions...................= ",In-
teractions)
    print("Number   of   Scattering   Events.............=
",Scattering)
    print("Number of Capture Events.................= ",Cap-
ture)
    print("Number of Fission Events.................= ",Fis-
sion)
    print("Number of Absorption Events.............= ",Ab-
sorption)
    print("Averega nu..............................= ",nu)
    print("Number of Neutrons Produced by Fission...= ",Neu-
trons_Produced)
    print("Number of Neutrons Leaked from System....= ",Leak-
age)
    print("Number of Neutrons Leaked into System....= ",Leak-
age)
    print("Effective    Multiplication    Factor(keff)....=
",keff)
    plt.show()

    return
```